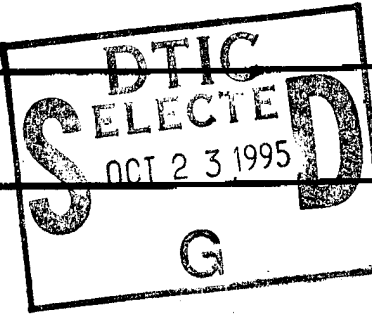# REPORT DOCUMENTATION PAGE

Form Approved.
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Sep 95	Final, Feb – Aug 95

4. TITLE AND SUBTITLE

Assembly Modeling Kernel
SBIR Phase I Final Report

5. FUNDING NUMBERS

C DAAH01-95-C-R035

6. AUTHOR(S)

Mike Webb

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Crystaliz Inc.
696 Virginia Road
Concord, MA 01742

8. PERFORMING ORGANIZATION REPORT NUMBER

AMK0002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency (DOD)
Defense SBIR Program

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DTIC SELECTED
OCT 2 3 1995
G

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

"A"

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report summarizes progress during PHase I of project Titled Assembly Modeling Kernel. The objectives explored during Phase I were: 1) Representations that span conceptual designs and physical constraints that drive decisions for electro-mechanical assemvlies 2) Concurrent evluation of design for manufacturability, assembly, and affordability, 3) Content based retrieval and indexing of product designs, and 4) WWW based approaches to supporting collaboration.

A software prottype that implemented free space computations, combinatoric analysis, semantic indexing through alternative key modeling, and a document control system that worked across WWW was implemented.

14. SUBJECT TERMS

Concurrent engineering, conceptual design

15. NUMBER OF PAGES

31

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

19951019 140

DTIC QUALITY INSPECTED 8

Assembly Modeling Kernel - SBIR Phase I Report

Sep. 28, 95

Sponsored By
Advanced Research Projects Agency (DOD)
Defense Small Business Innovation Research Program
ARPA Order No. 5916, Amdt 60

Issued by U.S. Army Missile Command Under

Contract #

Mike Webb, Principal Investigator
P. No: 508 287 4511, F. No: 508 287 4512

Effective Date of Contract: Feb 28, 1995

Contract Expiration Date: Aug. 28, 1995

Reporting Period: March - Aug. 1995

Crystaliz Inc., 696 Virginia Road, Concord, MA 01742

Executive Summary

This report summarizes the progress made during SBIR Phase I project titled **Assembly Modeling Kernel**. The Phase I project developed enabling technologies that supported several related objectives, which as a whole produces a systemic effect that is more than the sum of its parts.

The objectives explored during Phase I were:

1. Representations that span conceptual designs and physical laws /constraints that drive decisions

2. Concurrent evaluation of design for manufacturability, assembly, and affordability

3. Content and context based retrieval and indexing of product designs

4. World Wide Web based approaches for creating an information infrastructure for collaborative design

Together, achievement of these objectives will result in a design environment for assemblies that will provide order of magnitude improvement in modeling, simulation, engineering, and manufacturing of assemblies by a cross functional team that is not co-located.

First, we developed a class lattice that unified solid models with assembly models and kinematic models (kinematic constraints). Based on this unified model, algorithms were implemented that analyzed the configuration space (C-space) of a mechanism. A variety of computational geometry functions associated with the class lattice were implemented and tested, providing the computational base for functionalities related to C-space analysis. This work provided us with a representation that unified conceptual designs and physical constraints.

Second, in order to support concurrent evaluation, algorithms were also designed and implemented for evaluating alternative sequences of assembly operations and for performing dis-assembly analysis to test assemblability of products. Furthermore, design work towards assembly tolerancing using the above mentioned computational geometry functions was also performed.

Third, alternative key modeling which allows indexing and retrieval of product designs based on group technology classification was investigated. A class system which allowed *mixing-in* of appropriate categorization keys was implemented.

Finally, using Knowledge Query Manipulation Language (KQML), WWW, and a public domain Object Oriented Database (OODB), we explored the development of a

collaborative infrastructure that worked across WWW. A significant portion of the work in this direction was performed as part of another SBIR Phase II project. However, as part of this Phase I project, we explored the design of down-loadable applets that will allow interactive interaction across WWW for complex 3D information.

The work performed during Phase I provides a strong foundation for developing a collaborative, design, simulation, and engineering environment for complex assemblies that works across WWW. We intend to submit a Phase II proposal that completes this vision and develops a beta quality product.

Accesion For		
NTIS CRA&I		☒
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

INTRODUCTION

This report summarizes the progress made during the six-month funding period of SBIR Phase I project Assembly Modeling Kernel. After examining the feasibility of our various project goals, we opted to concentrate on the development of innovative data structures related to assembly-centered modeling and data modeling, within the framework of concurrent process design. During Phase One, we designed the classes and algorithms required for the representation and analysis of assemblies, including algorithms for analyzing an assembly's free space, and to support the combinatoric evaluation of alternative assembly sequences. Functions concerned with spatial analysis and computational geometry were implemented and tested, providing a computational basis for the geometric analysis of free space, disassembly analysis based on geometric feasibility, and assembly tolerancing based on variational modeling. We also investigated the development of a WWW based data modeling facility to support the concurrent design and analysis of assembly and manufacturing processes. Our primary focus in Phase Two will be to develop an extensible application framework devoted to the design and analysis of mechanisms and assembly processes. The assembly-centered functionalities of the kernel will include free space modeling, combinatorics modeling, data modeling, and assembly tolerancing.

Motivated by the lack of assembly-related capabilities among existing product modelers, an early goal of this project was to develop a part modeler whose functionalities included variational geometric modeling. However, the adequacy of existing modelers outside of the domain of assembly, as well as the time expenditure that would be required, led us to conclude that our efforts would be better spent outside the realm of part modeling, per se, concentrating instead on the development of assembly centered modeling tools capable of interfacing with an ``anonymous" part modeler. To support interaction with a generic solid modeler, we developed a class lattice for representing modelers, solids, surface features, and geometric entities.

Other functionalities examined during Phase One include:

1. *alternative key modeling,* which allows the user to access parts and assemblies with keys other than the name of the part or assembly and which facilitates the classification of assemblies into group technology categorizations

2. *physical connection modeling*, permitting the generation of a bond graph to simulate power flow between the parts of an assembly
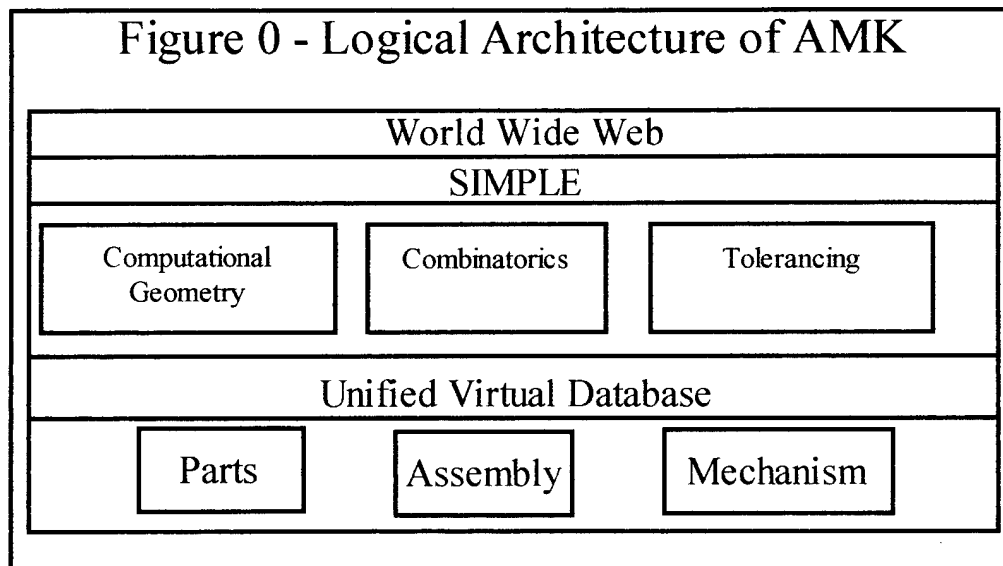
We opted not to peruse physical connection modeling, since power flow dynamics must be specified by the user, and it is unclear how to standardize that input in a manner that will be meaningful for a wide variety of mechanisms and types of physical connections. The use of medial-axis transforms for generating finite-

element meshes lies in the realm of solid modeling, which is outside of our chosen domain of assembly modeling.

PROJECT GOALS

The primary goal of this project is to develop assembly-centered modeling tools to support the design and analysis of mechanisms and assembly processes by a group of individuals. An extensible kernel was prototyped to facilitate the C-space analysis and assembly tolerancing of mechanisms, as well as combinatorial modeling and evaluation of alternative assembly sequences. Innovative data structures are being developed to support the concurrent engineering of mechanical parts and assemblies, within the cross-functional domains of parts modeling, mechanism design, and assembly process planning. A data modeling functionality is also being developed, to manage user-defined relationships across the cross-functional domains, with an added capability of alternative key modeling, enabling users to access parts and assemblies with keys based on design, engineering, or process knowledge over WWW.   In addition to facilitating mechanism design, the data modeling facility will support the comparative evaluation of alternative assembly plans and processes.

Our objective is to deliver, at the end of Phase II an extensible framework that provides an order of magnitude improvement in concurrent development of assemblies of mechanical products. Figure 0 shows an architectural overview of this

Figure 0 - Logical Architecture of AMK

World Wide Web		
SIMPLE		
Computational Geometry	Combinatorics	Tolerancing
Unified Virtual Database		
Parts	Assembly	Mechanism

framework.   During Phase I we prototyped all aspects of this framework excluding implementation of the tolerancing algorithms. In the following chapters we describe each of the functionalities.

Assembly-Centered Modeling

To fill the need for assembly-related capabilities not currently provided by geometric modelers, we seek to augment the functionalities of current modelers with the assembly centered capabilities of C-space analysis, combinatorial modeling of assembly plans, and assembly tolerancing. The assembly-based modeling tools will be designed to interface with ``anonymous'' part modelers, allowing a broad range of users to exploit our assembly-centered functionalities. Computational primitives will be provided to create assemblies through generic modeling operations, and to analyze and modify the assembly models within the framework of data modeling.

Mechanical assembly model may be constructed in the form of a kinematic tree, whose *joints* might represent either physical joints (prismatic, planar, rotary, universal, or gimbal) , *liaisons* denoting fixed relationships between fastened parts, or artificial pose constraints imposed by the user, to restrict a part's motion to a specified trajectory The varied semantics of the *joint* make possible a unified representation of kinematic mechanisms, assemblies composed of fastened subassemblies, and assembly motion trajectories. Accordingly, the class lattice for representing jointed assemblies simultaneously supports mechanism design and analysis, combinatorial analysis of alternative assembly sequences, and the design of assembly operations and workspaces. These functionalities augment the capabilities of existing solid modelers, whose assembly related functionalities are relatively limited.

C-space Analysis

The analysis of the free space about a mechanism yields the set of internal configurations and/or external poses that can be attained by the mechanism. C-space analysis yields a *C-space tree*, which represents the subset of poses in joint and pose parameter space that the mechanisms can reach in the presence of physical or artificial pose constraints imposed on its degrees of freedom. The free space information produced through C-space analysis facilitates the design and analysis of mechanical assemblies and manufacturing workspaces, as well as the planning of mating trajectories for assembly operations. C-space analysis permits the mechanism designer to note the effects of his or her design decisions on the extent and quality of the mechanism's free space. Downstream activities related to assembly task design need to accommodate the mechanism's free space when planning paths to avoid obstacles and selecting fixtures to immobilize the assembly components.

In the context of our assembly-centered modeling goals, C-space analysis may be performed with respect to various types of assembly parameters, such as the pose of a subassembly relative to a reference frame or a second subassembly; the internal joint parameters of an assembly; or the tolerance variables associated with part dimensions and mechanism descriptors. While the total number of parameters

defining a C-space is limited by computational complexity considerations, assembly parameters of various types may be combined as desired, enabling the mechanism designer to observe the effects of various selected part dimensions and tolerances on the mechanism's free space, and allowing the assembly process designer to plan trajectories within the composite C-space of an assembly's pose and joint parameters. Moreover, the process designer may impose artificial constraints on some of the assembly's degrees of freedom (e.g., confining the assembly pose to a linear path, or confining a joint parameter to a fixed value or interval), while allowing others to vary throughout their domains.

The Puma robot in Figure 1 provides an example of a mechanism whose C-space may be analyzed and modified with respect to design decisions, such as the positioning and dimensions of the separate assembly components. Figure 1 illustrates an interference zone in the free space of the Puma's second joint (the ``elbow''), whose associated link is obstructed by the Puma's base connector. The interference produces a gap in the free space of the elbow, which varies with the positioning of the ancestral, first joint. The Puma's C-space may be visualized in the parameter space of both joints, as shown in Figure 2, whose 2-D grid illustrates their composite C-space. With the zeroth (base) joint held frozen at a constant angle, the lighter regions in the 2-D grid indicate the ``legal'' joint angle pairs for which no interference exists between the components of the Puma. The dark horizontal region represents the base connector's interference with joint one, as well as joint two. The left-to-right variation in the horizontal region arises from the variability of the elbow's interference zone, which grows and shrinks with the movement of joint one.

Figure 1. Interference zone in the free space of the Puma's second joint.
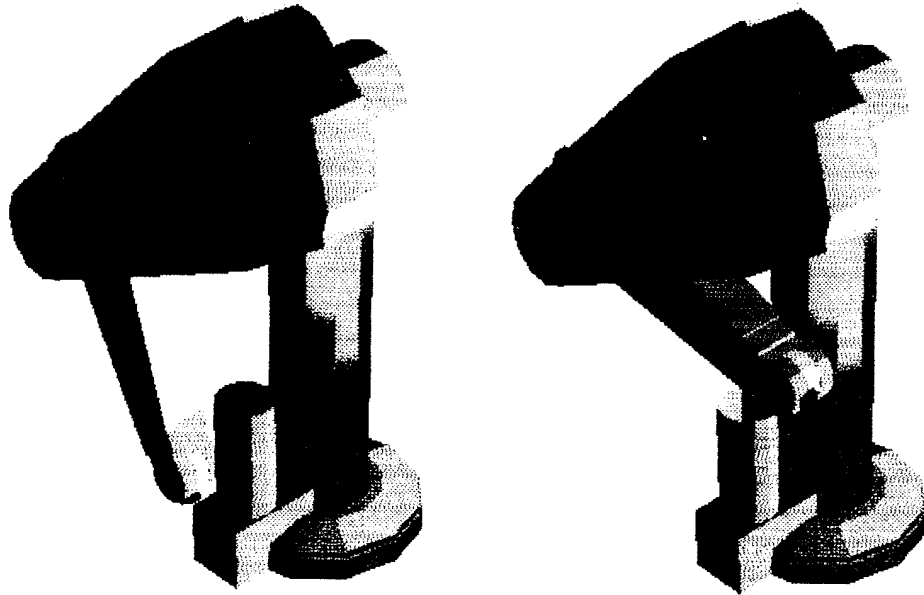
The dark vertical region represents a separate interference zone in the elbow's free space, arising from interference with vertical ``shoulder'' link, which is situated on top of the base.

The presence of the C-space obstacles depicted in Figure 2 suggest the need for a design modification in the Puma mechanism, such as flipping the robot's ``forearm''

**Figure 2. C-space of the Puma's first two joints.**
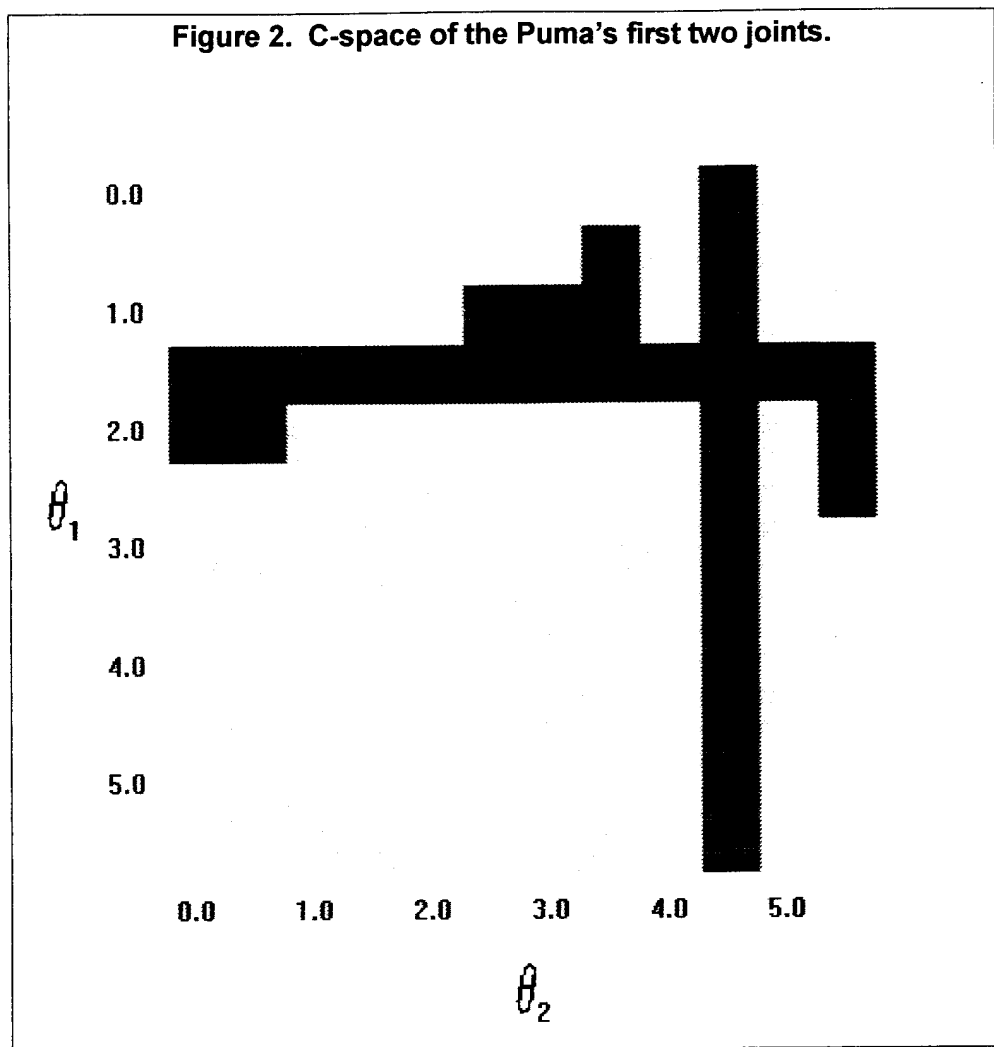
about its elbow, yielding the modified design shown in Figure 3.

This qualitative design decision eliminates much of the interference between the links of the Puma, in particular the interference between the

forearm and the rest of the Puma. The resulting C-space retains only a single interference zone, where the first joint's free space is still occluded by the base connector. This simple example illustrates the utility of the C-space analysis functionality as an aid in *qualitative* decision-making during the design of mechanical assembly models.

Figure 3. The C-space produced by flipping the forearm about the elbow.
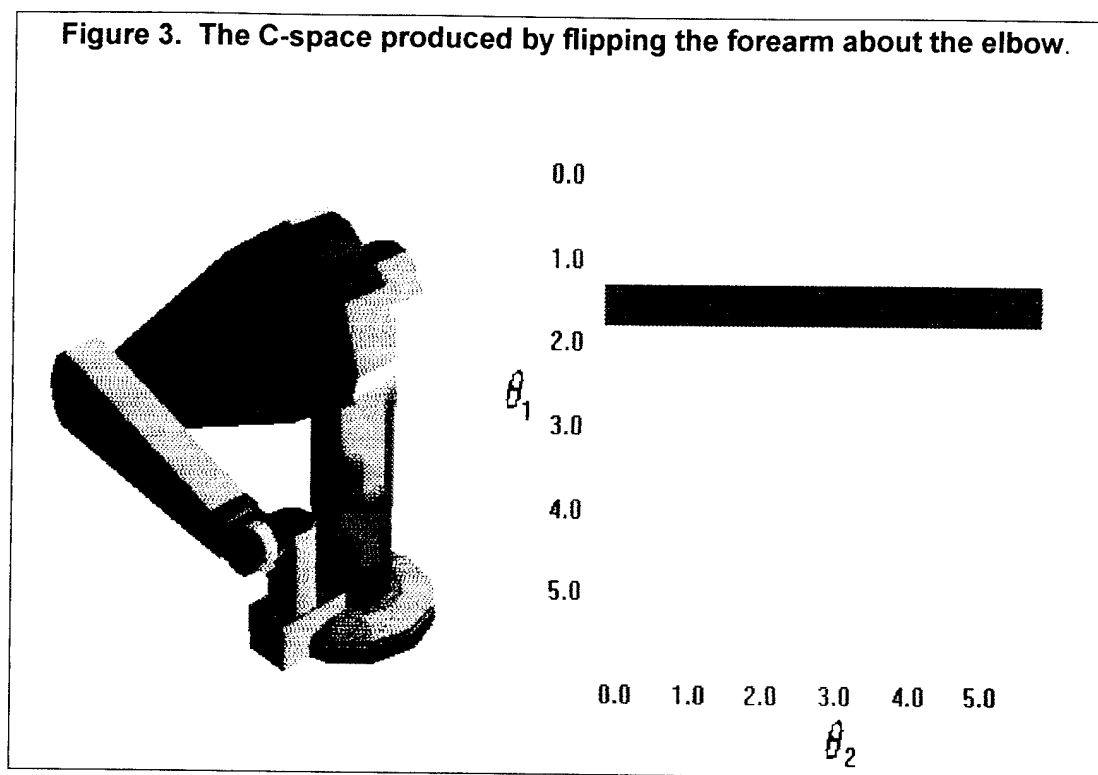
To demonstrate the role of C-space analysis in *quantitative* decision-making related to model dimensions, we refer to Figure 4. As revealed in the associated C-space grids, the various choices of Puma base length produce varying degrees of interference between the base connector and the upper and lower arms of the Puma. The extent and nature of a mechanism's free space is an important consideration to address during the design process, and our C-space functionality will serve to facilitate the designer's grasp of the relationship between the design variables and the C-space characteristics of the mechanism.

Figure 4. C-spaces associated with increasing base length
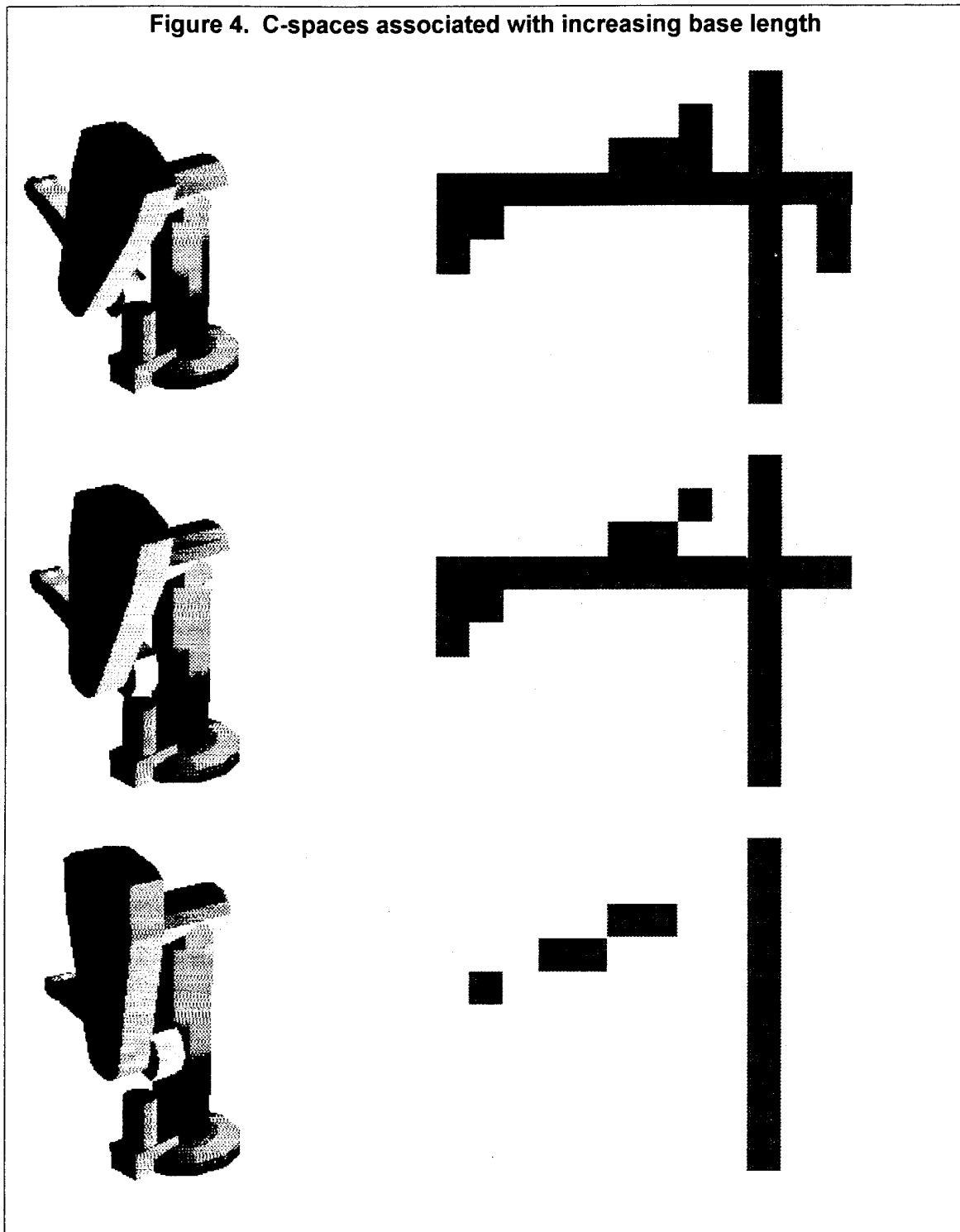
Combinatorial Analysis

The portion of the class lattice related to assembly modeling facilitates the construction of assembly models comprised of solids joined together by *liaison* joints. Each liaison represents the attachment of one solid to another, denoting either a rigid, fastened relationship, such as a pair of glued surfaces, or a non-rigid attachment, such as a loose insertion. A mechanism tree constructed with "liaison" joints may be decomposed into a variety of alternative subassemblies, depending on the order in which the liaisons are severed. Due to obstructing surfaces or fastener inaccessibility, some liaisons must be joined (or severed) before others. To support the enumeration and evaluation of alternative assembly plans, feasible liaison orderings are represented in an AND/OR graph[1], which represents all the alternative orders in which the liaisons may be severed. Each sequence of liaison severing operations represented in the graph corresponds to a feasible ordering of assembly operations, in reverse.   Given an AND/OR graph representation of alternative assembly operation sequences, the set of alternative assembly plans may be explored by traversing the different hyperarcs (OR's) in the graph.

The combinatorics functionality of the assembly-centered modeling kernel will rely on a graph-pruning process in which infeasible orderings of liaison severing operations are identified and removed from the AND/OR graph.   Traditional criteria for evaluating the feasibility of severing a liaison include geometric and mechanical feasibility, stability of the resulting subassemblies, and minimizing the number of directions from which assembly operations need to be performed. We have focused on the criteria of geometric and mechanical feasibility, which are at once the most crucial, and the most computationally challenging, criteria. Geometric feasibility is concerned with the separability of two subassemblies along a trajectory, and mechanical feasibility concerns accessibility of the fastener(s) associated with a liaison. Geometric feasibility is evaluated in two stages:

1. A local mobility procedure searches for contacting surfaces on the two subassemblies and determines if the surface relationships preclude relative motion along the given trajectory. This procedure can be performed rapidly, ruling out many liaison severing operations without proceeding to the global stage.

2. A global separability test is performed along the given trajectory to detect interference between the two subassemblies.

To illustrate the use of geometric feasibility analysis in assembly planning, we refer to the alternator assembly in Figure 5.

---

[1] Homem de Mello, L.S., Sanderson, A.C. (1989), "A correct and complete algorithm for the generation of mechanical assembly sequences", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 56-61.

Figure 5. An alternator assembly and its components.

A

C

B

D

E

F
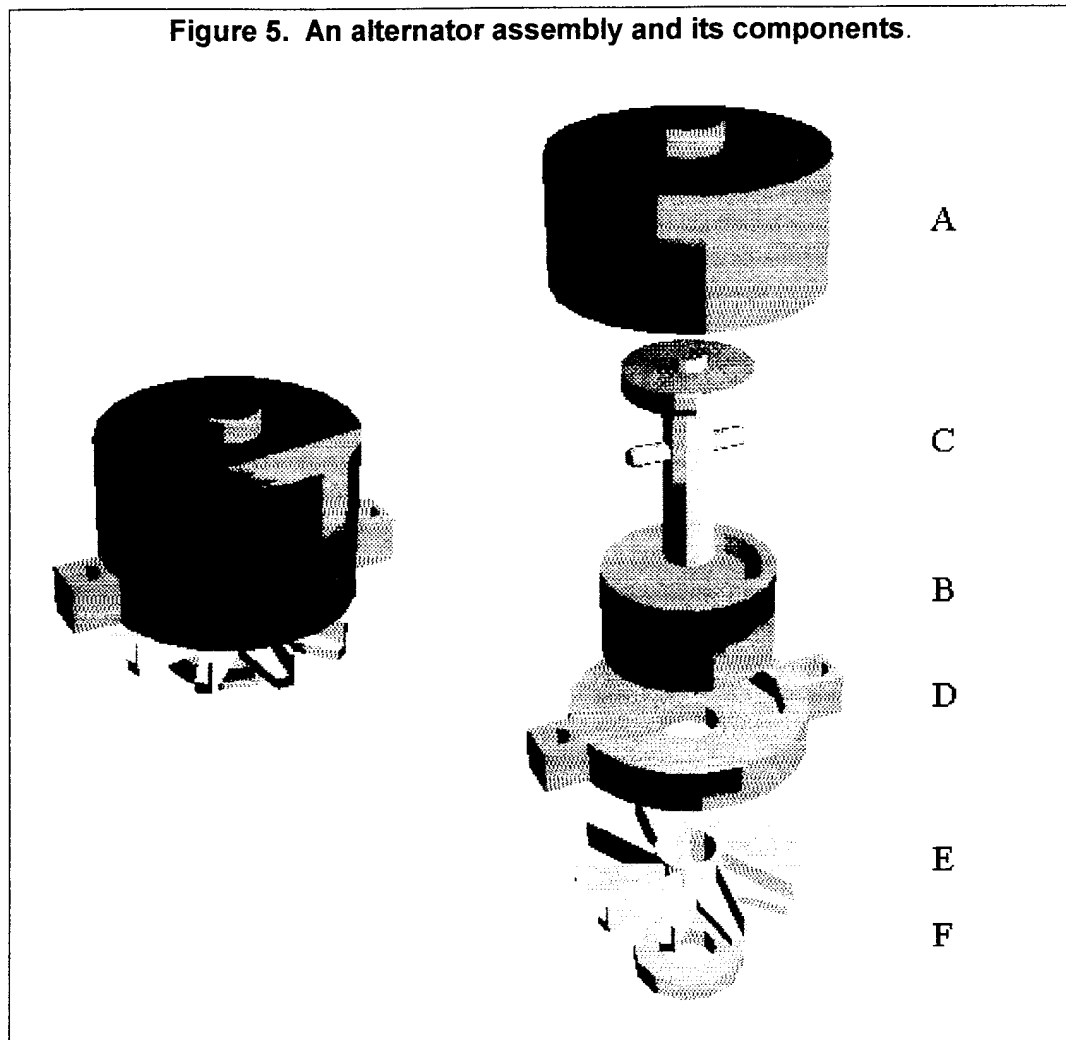
This assembly is composed of five parts, which are connected via five liaisons. Each liaison connects a pair of parts, so the five liaisons may be denoted by five part pairs: A-D, B-C, C-D, C-E, and C-F. In this particular assembly, all of the parts connect to the rotor shaft (C), with the exception of the rear housing (A), which attatches to the front housing (D). Initially, only two liaisons may be severed in the fully-assembled alternator: A-D or C-F. Liaison C-E, which involves the rotor shaft and the fan, cannot be severed before liaison C-F, since the pulley (F) blocks the path of the fan. Moreover, geometric feasibility testing reveals that the parts connected to the rotor shaft must be disassembled in this order: F, E, D, B. While this fact is readily apparent by simple inspection, liaison orderings are not so obvious

in complex assemblies, underscoring the need for automatic enumeration and evaluation of feasible assembly sequences.

A feasible disassembly sequence for the alternator assembly is shown in figure 6.

Figure 6. A feasible disassembly sequence
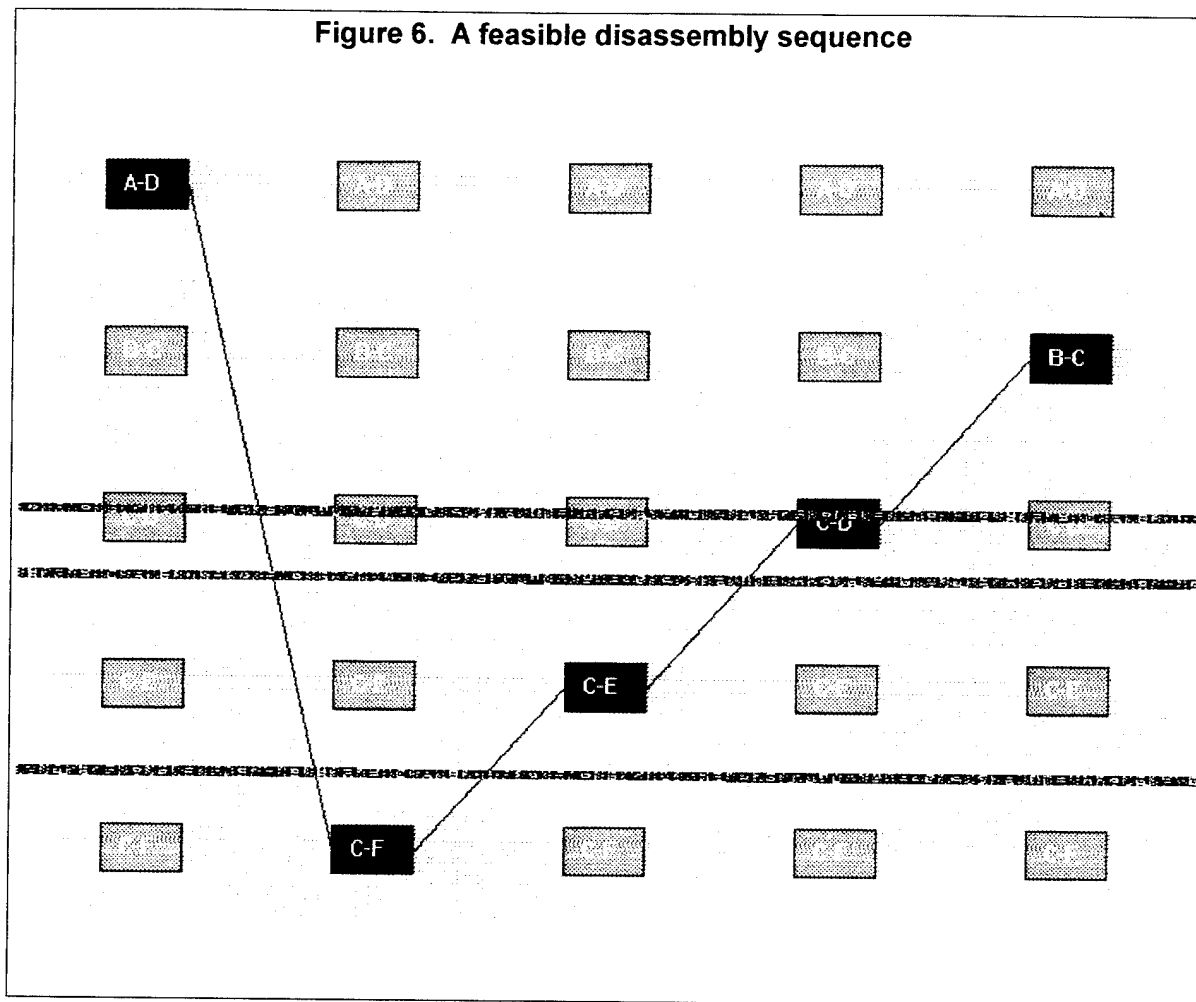
In this sequence, the rear housing is removed first, followed by the parts connected to the rotor shaft, which are removed in the required order stated above. Figure 7 shows all feasible paths *en masse*. The paths in Figure 7 differ only in their placement of liaison A-D in the sequence. The partial ordering of liaisons is computed through geometric feasibility testing of different disassembly operations in the various states of disassembly.

Figure 7. All feasible disassembly sequences
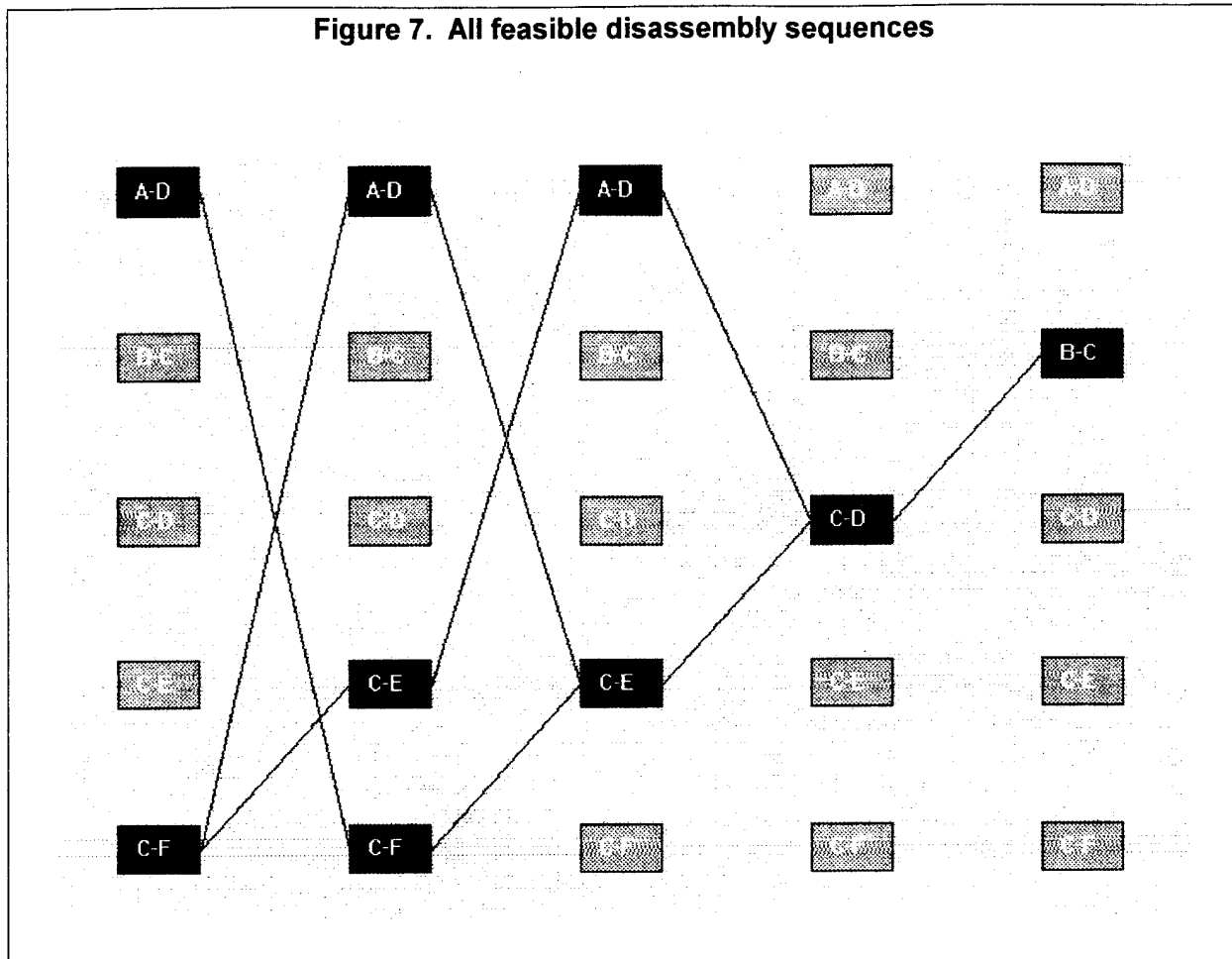
Geometric tolerancing

The assembly-centered modeling kernel will also support the modeling of tolerances in assemblies. Assembly tolerancing involves the calculation of part poses or pose uncertainy zones for the toleranced parts in an assembly. Since the part deformations allowed by part tolerance specifications usually prevent the assembled parts from being placed in their nominal configurations and stated surface relationships, feasible assembly part positions are analyzed to derive the expected or worst-case assembly configurations. Analysis of the toleranced parts' assembled positions facilitates the evaluation of design functions, helping to determine if the assembly model satisfies the design requirements.

Assembly tolerancing involves the derivation of pose parameter constraints, i.e., halfspace inequalities associated with nonoverlapping surface features. The pose constraints are calculated by propagating dimensional and geometric tolerance constraints to pose space. Given a set of tolerance specifications for the dimensions and geometries of the parts, the relative positions of the parts are either confined to uncertainty zones, estimated heuristically, or derived algebraically.

In Fleming's[2] algebraic approach to positioning toleranced assembly parts, the pose and feature constraints associated with each part in the assembly are expressed as symbolic equalities and inequalities. Extreme configurations of the contacts are derived from part geometries and tolerance bounds, producing symbolic constraints on the part positions. The resulting expressions are simplified to yield the relative positions of the parts in the assembly, or at least algebraic bounds on their uncertain poses. The approach used by Sodhi and Turner[3] relies on heurstics and linear programming to estimate the relative poses of the assembled parts, given a set of worst-case part deformation values. To estimate the assembly configuration associated with the specified part deformations, the assembled parts are assumed to fall together into relative positions that approximate their intended contact relationships as closely as possible. Using linear programming to optimize the proximity of nominally-contacting parts and surfaces, the assembled part poses are derived within a feasibility region defined by the surface contact constraints.

To facilitate the design and analysis of assemblies comprised of geometrically toleranced parts, the assembly modeling kernel will support the enumeration of the pose constraints which arise from contact relationships in an assembly. The pose constraints provided by the kernel may be used to compute tolerance stack-up in an assembly. The automatic generation of pose inequalities will be accomplished through the development of spatial analysis algorithms that identify interacting pairs of surface features on adjacent parts in an assembly model. The relevant feature pairs are those which constraint a single degree of freedom in the relative poses of the two parts (for example, vertex-face or edge-edge contacts between polyhedral parts) Each pose constraint represents a nonlinear halfspace in 6-dimensional pose space, characterized by a 5-dimensional C-surface. The pose constraints thus generated may be treated as symbolic or numerical expressions in any

[2] Fleming, A.D. (1989), ``A representation for geometrically toleranced parts'', in Woodwark, J. (Ed.) Geometric Reasoning, Clarendon Press, U. K., pp. 141-167.
[3] Sodhi, R., Turner, J.U. (1994), ``Relative positioning of variational part models for design analysis'', Computer-Aided Design, Vol. 26, No. 5, pp. 366-378.

assembly part positioning technique. In their nonlinear, algebraic form, the constraints provide pose inequalties for the symbolic algebra-based techniques Alternatively, each pose constraint surface may be linearized about its nominal part pose

Collaborative Information Infrastructure

Finally, an information infrastructure that will support collaborative, interactive browsing and update of a common database of assembly related data over WWW is key for the above functionalities to succeed. Several functionalities together provide support for content based indexing, querying, and update. First, content based indexing is supported through an object system that allows *mixing-in* of alternate keys to a base class. Using this functionality, design, analysis, and engineering data can be indexed according to their properties as well as other indexing schemes such as group technology. Second, use of Knowledge Query Manipulation Language (KQML) supports asynchronous querying and update of the common repository. Third, integration of KQML over HTTP allows users from various locations to issues queries and get results. Integration of KQML over the email infrastructure allows users to get notified when items they are interested in change. Fourth, a down-loadable view applet that supports 2D and 3D browsing (3D camera movement - zoom, pan, etc.) of 2D and 3D graphical information over WWW, a relationship facility that can be used to update multiple views when dependent views change, an object system in combination with KQML which allows dynamic creation of views, and ability to generate HTML on the fly allows the users to interact with the common repository of assembly related data over WWW.

PHASE ONE PROGRESS

During Phase One, we implemented a dynamic class lattice for representing generic geometric modelers, solids, assemblies, and kinematic mechanisms and for mixing-in content based indexing schemes. We also developed algorithms related to free space analysis, combinatorial enumeration of assembly sequences, and geometric feasibility of assembly operations. These data structures and algorithms were implemented in the functional programming language SIMPLE, which implements MIT Scheme, a meta class oriented object system, forward and backward chaining rules, and KQML. The algorithms were tested to verify their correctness, producing the C-spaces and assembly sequences that are illustrated in this report.

In addition, we prototyped a WWW based repository, dynamic creation of HTML forms, and down-loadable 3D browsers. Finally, these activities leveraged development efforts in other SBIR projects using which we developed SIMPLE.

Class lattice

A class lattice was developed for representing modelers, solids, and mechanism trees. Implemented in the functional programming language SIMPLE, classes and functions were designed for representing and manipulating geometric entities, providing the essential functionalities, such as distance calculation, required by the free space , tolerancing, and combinatorics algorithms. Classes including *modeler*, *model, face, edge*, etc. were created to support interaction with a generic geometric modeler, along with corresponding ``mix-in" classes pertaining to the Noodles modeler (obtained from Carnegie Mellon University). Classes supporting the representation of assemblies as mechanism trees include "link", which relates an assembly component to a joint, and "joint", which can represent a mechanical joint (prismatic, planar, rotary, universal, or gimbal) or a "liaison", i.e., a fixed relationship between two fastened parts. As implemented in SIMPLE, these classes and their associated methods provide a basis for the development and testing of algorithms for generating the discretized free space of a mechanical assembly, and for evaluating the geometric feasibility of alternative assembly sequences.

The SIMPLE language supports mixing-in of alternative indexing schemes and we designed a group technology based indexing scheme that took advantage of this capability.

Utility functions for spatial analysis

Utility functions associated with spatial analysis were designed and implemented to support the assembly-centered functionalities of C-space analysis, disassembly analysis, and assembly tolerancing. The spatial analysis algorithms are concerned with computing distances, performing orthogonal projections, and testing point set

inclusion, as required by the higher-level assembly-centered algorithms. These functions are primarily associated with the *line* and *plane* classes (see Appendix) A higher-level distance calculation procedure, which computes distances between solid models, relies on utility functions associated with the *model, face, edge,* and *vertex* classes.

Relying on orthogonal projections and half-space predicates, these utility functions verify that the closest points on a pair of solids lie on a specified pair of surface features. The motivation behind such utility functions is as follows.

The C-space tree construction algorithm calls for repetitive distance calculations, performed iteratively for incrementally increasing joint parameter values, to detect possible interference between a mechanism link and an obstacle. Potentially time intensive, the distance computations required by the C-space procedure were made more efficient by adopting a technique described by Lin and Canny[4], which exploits the predictability of the closest surface features involved in successive distance calculations. The distance measurement procedure applies only to convex solids, so the original solids must be partitioned into convex primitives or replaced by a convex bounding box Measuring the distance between two convex solids involves first identifying the closest pair of surface features (vertices, edges, faces) on the two solids, then measuring the distance between that pair of features. Lacking any a priori knowledge of the closest features, the distance calculation procedure must process all relevant pairs of features to find the closest pair. By recalling the closest pair of features already computed for similar configurations, however, the closest features can usually be predicted, or at least found to be located nearby the predicted pair. Indeed, as the algorithm incrementally steps through a joint's parameter range, the distance from a swept link to an obstacle tends to involve the same pair of closest features. Consequently, distance calculations are expedited by keeping track of the ongoing pair of closest features, which are rapidly verified or updated during each iteration of the C-space algorithm. Since the closest pair of features tends to remain the same from one iteration to the next, and only a limited number of inequality constraints need to be verified, the distances are arguably computed in constant time.

To support the efficient, repetitive calculation of distances between convex primitive shapes, special functions were developed for the *face, edge,* and *vertex* classes to verify a point's satisfaction of a set of half-space constraints associated with a face, edge, or vertex. If the point satisfies all of a given feature's half-space constraints, then the feature is verified to be the closest feature on the solid to the point in question. If a half-space constraint is not satisfied, the algorithm ``walks'' to the adjacent feature associated with the violated constraint, and repeats the verification

---

[4] Lin, M.C., Canny, J.F. (1991), ``A fast algorithm for incremental distance calculation'', *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1008-1014.

process recursively, until the closest feature has been identified. This process is performed for both solids, yielding their closest features and points.

C-space analysis

An algorithm was designed to construct a discretized representation of a mechanical assembly's free space. The algorithm generates the C-space of reachable, non overlapping configurations that can be attained by a set of spatially or mechanically related solids. A preliminary implementation of the algorithm was accomplished during Phase One, to demonstrate the correctness and feasibility of the C-space construction procedure. The C-spaces generated by the algorithm can facilitate the design of mechanisms, industrial workspaces, and trajectories for assembly operations. Examples of C-spaces generated by the algorithm to facilitate mechanism design appear earlier in this report (see Figures 2, 3, and 4).

Figure 8 demonstrates how C-space analysis may also facilitate path planning for assembly operations.

Figure 8: Path Planning Example
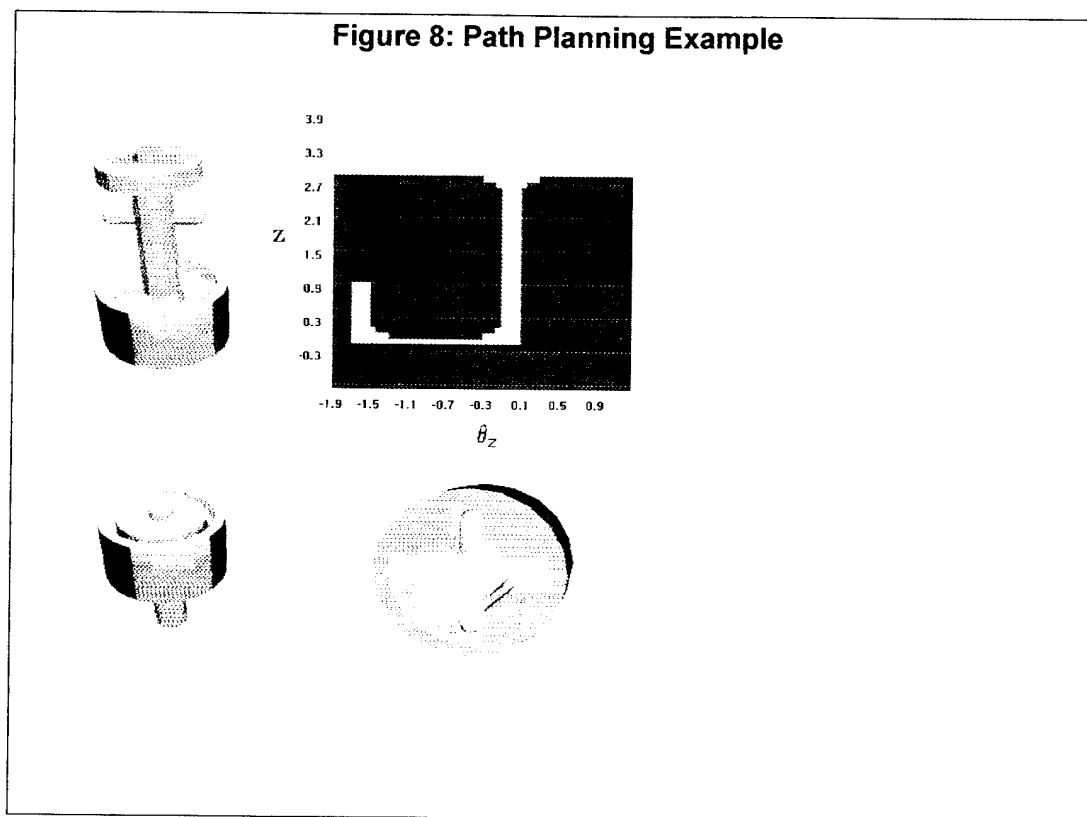
The rotor subassembly in Figure 8 consists of parts B and C of the alternator assembly shown in Figure 5. The rotor shaft is assembled to the rotor wheel by way

of an insert-and-twist operation, involving three sequential motions of the rotor shaft: (1) a translational insertion into the slotted hole of the wheel, during which the key of the shaft must be aligned with the slot, until the key has cleared the slot; (2) a 90-degree rotation, to align the key with a second slot on the underside of the wheel; and (3) a translation in the reverse direction, bringing the key to its final position inside the second slot. Also shown in Figure 8 is the rotor assembly's Cspace, which is defined in the parameter space of the translational and rotational motions just described. The C-space has three main channels, corresponding to the three assembly path segments. The width of each channel indicates the leeway in the trajectory segment. The channels tend to widen at the points of transition to the next channel. This phenomenon, which the C-space analysis quantifies for path-planning purposes, reflects the varying amount of angular restriction placed upon the rotor shaft as the key starts to enter or leave a wheel slot.

Our C-space construction algorithm combines the approaches of Handey[5] and ACT[6] to construct a C-space tree representing the free space range of an ordered sequence of joints in a mechanical assembly. Our version of the algorithm generalizes the domain of Handey to consist of multiple chains of linkages, or independent parts or subassemblies, as in Figure 8. Following Handey, the C-space algorithm discretizes each joint parameter range into cells labeled FULL or EMPTY (free), recursively expanding the EMPTY nodes in the tree to the next level. Following the approach of ACT, the EMPTY nodes are divided into P/EMPTY (possibly empty) or C/EMPTY (certainly empty) nodes, using swept volume operations to determine if the entire subtree below an EMPTY node is free of interference and need not be expanded. The current design supports the creation of virtual mechanisms by including "model" class methods for generating swept volumes along, or about, a given translational or rotational axis.

The range of interference between two solids is computed using Handey's method of merging the individual interference intervals of all edge-face pairs on the solids. The advantage to this approach is that concavities in the solids do not complicate the interference detection procedure. The disadvantage is that the number of edge-face pairs is quadratic in the number of edges and faces on the two solids. To mitigate the complexity of Handey's interference detection procedure, the algorithm's efficiency is improved by adopting techniques from ACT and elsewhere, such as the monitoring of distance to the bounding volumes of potential obstacles, and the efficient measurement of distances between the convex components of solids.

[5] Lozano-Perez,, T., Jones, J.L, Mazer, E., O'Donnel, P.A. (1992), *Handey -- A Robot Task Planner*, MIT Press, Cambridge, Mass.

[6] Mazer, E., Pertin-Troccaz, J., Lefevre, J.-M., Faverjon, B., Ijel, A., Bellier, C., Ferrari, B., Barret, M., Sellers, P., Lefebvre, J.-M.,, Hassoun, M., Alchami (1991), O., ``ACT: a robot programming environment", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1427-32.

To compute the free space interval(s) for a link in the mechanism tree, the link's interference interval with respect to each potential obstacle must be calculated. The detailed calculation of interference intervals, which involves the pairing of all edges and faces on the two solids, can be avoided whenever the sweeping link's closest distance to the obstacle is greater than zero. Following the technique used in ACT, these distance measurements are performed upon the models' bounding volumes, which are topologically simpler than the models themselves. The measured distances between a link and the bounding volumes of potential obstacles are recorded for posterity, allowing future iterations of collision detection operations be omitted when performed for nearby assembly configurations. As described in the previous section, the distance measurements themselves are performed through an efficient technique in which the closest features on the two solids are continuously monitored. Once the closest features on the incrementally moving solids have been identified as part of the first distance measurement procedure, the subsequent identification of closest features during successive measurements requires only the verification of a few half-space constraints. The verification or updating of closest features requires a computation time that is minimal and does not vary the number of surface features on the solids.

Combinatorial analysis

Algorithms have also been designed to determine the geometric feasibility of assembly operation sequences. A mechanism tree constructed with "liaison" joints may be decomposed into a variety of alternative subassemblies, depending on the order in which the liaisons are severed. Feasible liaison orderings are represented in an AND/OR graph, which contains a partial ordering of the liaison severing operations. Traditional criteria for evaluating the feasibility of severing a liaison include geometric and mechanical feasibility, stability of the resulting subassemblies, and minimizing the number of directions from which assembly operations need to be performed. We have focused on the criteria of geometric and mechanical feasibility, which are at once the most crucial, and the most computationally challenging, criteria. Geometric feasibility is concerned with the separability of two subassemblies along a trajectory, and mechanical feasibility concerns accessibility of the fastener(s) associated with a liaison.

To test the geometric feasibility of severing a liaison, the algorithm must be provided with a path direction, along which one subassembly separates from the other subassembly. The separation path, which is the reverse of the actual mating trajectory, may be supplied a priori or computed from contact geometry and heuristics. The geometric feasibility test is performed in two stages: (1) A local mobility procedure searches for contacting surfaces on the two subassemblies and determines if the surface relationships preclude relative motion along the given trajectory. This procedure can be performed rapidly, ruling out many liaison severing operations without proceeding to the global stage. (2) A global separability test is

performed along the given trajectory to detect interference between the two subassemblies. Employing a graphical projection technique,[7] global separability is determined by finding the opposing faces on the two subassemblies' solid models, projecting the faces onto a plane, and detecting a non null intersection between the projected faces. If one projected face falls entirely within the borders of the other, the algorithm must also check to see if the former face lies entirely within a hole of the latter, which would indicate a null intersection.

Once the geometric feasibility of separating a pair of subassemblies has been verified, the mechanical feasibility must also be tested, to decide whether the fastener(s) associated with the liaison are accessible or blocked by adjacent parts. At a minimum, mechanical feasibility testing is performed by determining the geometric separability of the fastener from the assembly. If a model of the fastening tool is supplied, the tool's accessibility to the liaison site is also evaluated.

Information Infrastructure

We explored several aspects of the collaborative infrastructure as part of this project and as part of another SBIR Phase II project titled Constraint Modeling Kernel (CMK). CMK uses the Knowledge Query Manipulation Language (KQML) protocol to support distributed, asynchronous access, query, and update of a repository by multiple individuals. On top of this infrastructure, we implemented a way to generate HTML forms from a versioned repository, accept KQML queries entered using WWW clients, and designed down-loadable 3D viewing mechanisms.

---

[7] Chen, C.L.P, Wichman, C.A. (1993), ``A systematic approach for design and planning of mechanical assemblies'', *AIEDAM,* Vol 7, No. 1, pp. 19-36.

STATUS

The following chart summarizes our progress in Phase I against goals that were set out in the proposal that was funded.

Task	Specification	Design	Prototyping
Unified Representation - Class Lattice - Analysis Algorithms			
Concurrent Evaluation - Combinatorics			
Concurrent Evaluation - Tolerancing			
Content Based Retrieval - Alternate Key Indexing			
Content Based Retrieval - Group Technology Mixins			
Information Infrastructure - WWW integration - Content based asynchronous, query and update (KQML)			
Down-loadable applets			

PHASE TWO OBJECTIVES

Phase Two activities center around the development of an extensible framework to support assembly-centered modeling functionalities related to the design and analysis of mechanisms and assembly processes. The kernel will incorporate and augment existing products devoted to solid modeling and data management. Functionalities of the encompassing framework will include C-space analysis, combinatorial modeling and evaluation of alternative assembly sequences, assembly tolerancing, data management, alternative key modeling, and down-loadable 3D controls over WWW. During Phase Two, the class lattice designed during Phase One will be augmented to represent tolerances for solids and assemblies, and the assembly-centered modeling facilities will be structured within a data modeling framework. Activities planned for Phase Two include:

- Developing an interface to a data management program, to support data modeling of mechanisms and assembly process designs.

- Implementing a content based alternative key modeling facility, based group technology.

- Implementing navigation primitives to explore alternative paths in an AND/OR graph of alternative assembly sequences

- Augmenting the class lattice to incorporate tolerances in the dimensions of parts and assemblies, and developing tools to support the calculation of tolerance stack-up.

- Generalizing the C-space algorithm to include part and joint tolerance variables among the set of parameters which define a C-space.

- Implementing the numerically-intensive portions of the C-space and geometric feasibility algorithms in C, to improve their real-time performance.

- Implementing down-loadable 3D controls within WWW browsers

- Implementing a dynamic view creation mechanism that works across WWW

APPENDIX

Several of the classes designed for geometric modeling, spatial analysis, and representation of kinematic mechanisms are described below, in terms of their value and function slots. For the sake of brevity, only the representative function slots are listed.

Classes associated with geometric modeling

The modeler class

This class represents generic geometric modelers, whose capabilities include model creation

and duplication, as well as CSG operations and sweeping operations. The *modeler* class is intended to represent a modeler invoked by the C-space, combinatorial analysis, and tolerancing functionalities, to copy or manipulate existing models, or to generate new models representing the bounding boxes or swept volumes of existing models.

Function slots:

- **modeler-make-block** < *length, width, height* >

    Create a block model of the specified dimensions.

    Related function slots:

> modeler-make-cone,
> modeler-make-cylinder,
> modeler-make-ellipsoid
> modeler-make-torus

modeler-union-models < *model1, model2*>

Create a new model that is the union of two existing models (destructive).

Related function slots:

modeler-intersect-models

modeler-subtract-models

- modeler-make-translational-sweep <*points, distance*>

Create model by sweeping a set of points by a specified translational distance.

Related function slots:

modeler-make-rotational-sweep

The **model** class

This class represents 3-D geometric models and their topology. The *model* class is used by the assembly-centered modeling functionalities to represent and access the features of solid models contained in mechanism trees.

Value slots:

■   model-faces

A list of faces associated with the model.

Related value slots:

model-edges

model-vertices

■   model-volume

The real-valued volume of the model.

■   model-b-box-size

A vector containing the 3 size dimensions of the bounding box of the model.

Related value slots:

model-b-box-center

Function slots:

■   model-duplicate <.>

.Create an identical copy of this model.

■   model-translate < x, y, z>

Translate this model by the given displacement.

Related function slots:

model-rotate
model-transform

The **face** class

This class represents faces in geometric models.

Value slots:

- face-loops

    A list of loops associated with the face.

- face-normal

    The outward normal of the face, represented as a 3-element vector

Related classes:

loop
edge
vertex

Classes associated with spatial geometry

The **plane** class

This class represents planes in 3-space.

Value slots:

- plane-vector

    A 4-element vector containing the plane's normal and distance to the origin.

    Related accessors:

    plane-normal
    plane-scaler

Function slots:

- plane-reverse < >

Create a new plane whose normal is anti-parallel to this plane.

■ plane-nearest-point <vector>

Compute the projection of a point vector onto this plane.

plane-point-relation-plane <function, vector>

Return true if a point vector .satisfies a specified relationship with this plane.s

The **line** class

This class represents lines in 3-space.

Value slots:

■ line-start

A 3-element vector representing this line's first defining point.

Related slots:

line-end

Related accessors:

line-tangent

Function slots:

■ line-nearest-point <vector>

Compute the closest point on this line to a point vector.

■ line-nearest-points-line <line>

Compute the closest point on this line to another line and vice-versa.

Classes associated with assembly modeling

The assembly-component-mixin class

This class represents assembly components, such as subassemblies or parts. This class is added to the list of mixins of a given class, such as *model,* in order to endow the given class with assembly component attributes, such as a set of links.

Value slots:

- assembly-component-links

  .A list of links, each corresponding to a transformed joint.

The **link** class

This class represents half-joints in an assembly.

Value slots:

- link-component

  .The assembly component associated with this link.

- link-joint

  .The joint associated with this link.

- link-frame

  .The transformation from this link's assembly component and this link.

The **joint** class

This class represents joints, such as prismatic, rotary, or liaison joints, between the components of an assembly. The *joint* class is used to compose kinematic mechanisms and assemblies of geometrically-modeled parts.

Value slots:

- joint-links

  The pair of half-joints associated with this joint, which normally point to a pair of assembly components.

Descendent classes:

    revolute-joint
    prismatic-joint
    liaison-joint